Local Search Techniques For Educational Timetabling Problems

LOCAL SEARCH TECHNIQUES FOR EDUCATIONAL TIMETABLING PROBLEMS

Andrea Schaerf[1] and Luca Di Gaspero[2]

1. Dipartimento di Ingegneria Elettrica, Gestionale e Meccanica,

Università di Udine,

Via delle Scienze 208, 33100 Udine, ITALY

e-mail: `schaerf@uniud.it`

2. Dipartimento di Matematica e Informatica

Università di Udine,

Via delle Scienze 208, 33100 Udine, ITALY,

e-mail: `digasper@dimi.uniud.it`

Abstract

Local Search is an emerging paradigm for combinatorial search, which has been recently shown to be very effective for a large number of combinatorial problems.

Educational timetabling problems consist in scheduling a sequence of lectures or examinations between teachers and students in a prefixed period of time (typically a week), satisfying a set of constraints of various types. Problems in this family differ from each other based on the type of events and the kind of institution involved. A number of techniques based on local search have been proposed to address, with success, a variety of timetabling problems.

In this paper we present our work on the application of local search techniques to various versions of the educational timetabling problem. We also sketch some of the main results and the specialized software tool employed for the development of the algorithm.

Keywords: Timetabling, Local Search, Tabu Search

1 Introduction

Local search is a family of general-purpose techniques for search and optimization problems that are based on several variants of a simple iterative idea: Given a solution of a problem instance they try to fix it in order to obtain a better solution, and they iteratively continue until no improvement is possible anymore.

Local search techniques are *non-exhaustive*, in the sense that they do not guarantee to find a feasible (or optimal) solution, but they search non-systematically until a specific stop criterion is satisfied. Nevertheless, these techniques are very appealing because of their effectiveness, efficiency, and widespread applicability.

The most popular local search techniques are *Hill Climbing*, *Simulated Annealing*, and *Tabu Search* (see, e.g. [1]), which rely on three different philosophies: simple, probabilistic and memory-based respectively.

Educational timetabling problems consist in scheduling events (typically lectures or examinations) which involve instructors and students in a prefixed period of time (usually a week, or rarely two weeks). A considerable attention has been devoted to automated timetabling during the last forty years: starting with [18], many papers related to automated timetabling have been published in conferences proceedings and journals. In addition, several applications have been developed and employed with a good success.

The application of local search to educational timetabling problems is quite recent. The main papers on this subject belong to the early 90s [2, 19], and since then, the research on local search methods for timetabling has never stopped (see e.g., [3, 11, 12, 23]). Moreover, also many sessions of the conference series on timetabling problems, the PATAT series [5, 6, 9] [1], are devoted to local search methods.

In this paper, we introduce the local search paradigm (Section 2) and we describe the various versions of the educational timetabling problem (Section 3). In Section 4 we show our contributions on the field, placed within the relevant literature, for the solution of timetabling problems. Furthermore, in Section 5 we introduce our software tools for developing local search applications, namely the framework EasyLocal++. Finally, in Section 6, we draw some conclusions.

2 Local Search

The Local search paradigm is a family of general-purpose optimization techniques that can be described as follows. Given an instance p of a problem P, we associate a *search space* S with it. Each element $s \in S$ corresponds to a potential solution of p, and is called a *state* of p. Local search relies on a function $\mathcal{N}$ (depending on the structure of P) which assigns to each $s \in S$ its *neighborhood* $\mathcal{N}(s) \subseteq S$. Each state $s' \in \mathcal{N}(s)$ is called a *neighbor* of s. The neighborhood of a state is usually described in an intensional fashion, i.e., in terms of atomic local changes (called *moves*) that can be applied on it.

A local search algorithm starts from an initial state s_0 (which can be obtained with some other technique or generated randomly) and enters a loop that *navigates* the search space, stepping from one state s_i to one of its neighbors s_{i+1}. The search is driven by a *cost function* F that estimates the quality of each state. For constraint satisfaction problems, F generally accounts for the number of violated constraints, whereas for optimization problems it takes into account also the objective function of the problem. Although the selection of the moves is based on the cost function, the precise way such a selection takes place depends on the specific local search technique.

Two other important components of a local search algorithm are the choice of the initial solution and the stop criterion, which determines when the search phase is over and the best solution found is returned. However, also these issues are usually part of the specific local search technique.

2.1 Basic Local Search Techniques

The three most popular local search techniques proposed in the literature are Hill Climbing (HC), Simulated Annealing (SA) and Tabu Search (TS). In the following we only sketch the

[1]PATAT is actually focussed on timetabling in a broader sense, which includes, for example, employee timetabling and sport tournament timetabling.

main aspects of these techniques; the reader who is interested in a more detailed presentation can refer, for example, to [24].

HC relies on the simple idea of performing only moves which improve or leave unchanged the value of the cost function. The selection of the move is made either by randomly sampling a set of moves, or by exhaustively exploring the neighborhood looking for the best move (in the latter case the technique is also known as Steepest Descent). The search is stopped when no improvement can be found anymore, or when a fixed number of iterations from the last strict improvement has elapsed.

The main problem of this technique is the fact that it can easily get stuck in so-called *strict local minima* – i.e., states of the search space that are "surrounded" by worsening solutions – or it can cycle among a set of equally valued states (*plateaux*). To cope with this problem, several ideas which allow also worsening moves have been presented in the literature. Among those, we present SA and TS, which relies on two different philosophies: probabilistic decision and memory-based prohibition.

SA is a probabilistic local search technique whose name comes from the fact that it simulates the cooling of a collection of hot vibrating atoms. After a random initial solution is created, the algorithm generates a move m at each iteration, and it performs m according to the following rule: The move is certainly accepted if it improves the cost value of the current state, otherwise the move is accepted according to a time decreasing probability. In the latter case, if the acceptance test is negative then no move is performed, and the current state is left unchanged.

The control parameter of the probability distribution is called *temperature*. At the beginning of the search it is set at an appropriate high level and it is decreased during the search following a so-called *cooling scheme*. The whole search stops when the temperature reaches a value very close to zero, hence no worsening move could be accepted anymore. In this case we say that the system is *frozen*, and the solution obtained at that temperature is obviously a (hopefully good) local minimum.

TS is a method which keeps track of features of previous solutions in order to inhibit the search to visit them again. The basic mechanism of TS is quite simple: At each iteration a subset of the neighborhood of the current solution is explored, and the neighbor that gives the minimum value of the cost function becomes the new current solution independently of the fact that its value is better or worse than the value of the current solution. The subset to explore at each step is determined by a so-called *tabu list*, which is a short-term memory of moves that are forbidden to execute. These moves are not taken into account in the exploration of the neighborhood, because they could lead to already visited solutions.

Notice that keeping track of moves prevents the algorithm not only to visit previous solutions, but also states which shares some features with already visited solutions. For this reason, there is also a mechanism, called *aspiration*, that overrides the tabu status of a move if it gives a *large* improvement of the cost function

The TS procedure stops either when the number of iterations reaches a given value or when the value of the cost function in the current solution reaches a given lower bound.

2.2 Improvements on the Basic Techniques

One of the main issues of local search techniques is the way they deal with the local minima of the cost function. Even if the search procedure employs some specific mechanism for escaping them (like SA or TS), a local minimum still behaves as a sort of *attractor*. Intuitively, when a trajectory moves away from a local minimum and steps through a solution "near" to it, even though it is not allowed to go back to the minimum itself, it still tends to move "forward" instead of moving in an "opposite" direction[2].

For the above reason, the search procedure needs to use some form of *diversification* strategy that allows the search trajectories not only to escape a local minimum, but to move "far" from it thus avoiding this sort of *chaotic trapping* around the local minimum.

On the other hand, for practical problems the landscape of the objective function is usually such that the objective function is correlated in neighbors (and near) solutions. Therefore, once a good solution is found, it is reasonable to search in the proximity of it for a better one. For this reason, when a local minimum is reached the search should be in some way *intensified* around it.

In conclusion, the search algorithm should be able to balance the two sometimes conflicting objectives; it should diversify and intensify by moving outside the attraction area of already visited local minima, but not too far from it.

Several strategies have been proposed in the literature to solve this issue by giving the appropriate quantity of each in different phases of the search (see [17]).

One example of such a strategy is the *shifting penalty* (see, e.g., [16]), which is a mechanism that changes continuously the shape of the cost function in an adaptive manner. This way, it causes the local search algorithm to visit solutions that have a different structure than the previously visited ones. In details, the constraints which are satisfied for a given number of iterations will be relaxed in order to allow the exploration of solutions where those constraints do not hold. Whereas, if some constraint is not satisfied for a long time, it is tightened, with the aim of driving the search toward its satisfaction.

Another control knob that can be used specifically for the TS technique is the length of the tabu list. The *dynamic tabu list* approach adaptively modifies the tabu list length in the following way: if the sequence of the last performed moves is improving the cost function, then the tabu list length is shortened to intensify the search; otherwise, if a sequence of moves induces a degradation, the length of the tabu list is extended to escape from that region of the search space.

2.3 Composite Techniques

One attractive property of the local search paradigm is that different techniques can be combined and alternated to give rise to complex algorithms. An example of a simple mechanism for combining different techniques and/or different neighborhood relations is what we call the *token-ring* strategy: Given an initial state and a set of basic local search techniques, the token-ring search makes circularly a run of each technique, always starting from the best solution found by the previous one. The search stops when no improvement is obtained by any algorithm in a fixed number of rounds, whereas each component runner stops according to its specific criteria.

The effectiveness of token-ring search for two runners, called *tandem* search, has been stressed by several authors (see [17]). In particular, when one of the two runners is not used with the aim of improving the cost function, but rather for diversifying the search region, this idea falls under the name of *iterated* local search (see, e.g., [22]).

[2]We remark that here the quoted terms are used in an intuitive way, without referring to any metric or distance. A more formal discussion is given, for example, in [4]

3 Educational Timetabling Problems

The educational timetabling problem consists in scheduling a sequence of events (typically lectures or examinations) which involves teachers and students in a prefixed period of time, satisfying a set of constraints of various types. Constraints involve, among others, overlapping of events with common participants, capacity of rooms, and student and teacher workload.

A large number of variants of the timetabling problem have been proposed in the literature, which differ from each other based on the type of events, the kind of institution involved (university or school) and the type and the relative influence of constraints. According to these dimensions, timetabling problems are classified into three main classes described in the following subsections.

3.1 School timetabling

The school timetabling problem, also known as the class-teacher problem, regards the weekly scheduling for all the lectures of a school.

Given three sets of classes, teachers, and periods, respectively, and a requirement matrix that states the number of lectures that each teacher has to give to each class; the problem consists in assigning lectures to periods in such a way that no teacher or class is involved in more than one lecture at a time and each teacher gives the right number of lectures to each class.

Typical additional constraints regard teacher's day-off, joint lectures to two or more classes, co-presence of teachers, and compactness of class schedule. In particular, the compactness constraint states that classes cannot be uncovered for any period, but the first or the last of the day. Compactness is very crucial for school timetabling, and it represents one of the major differences between school timetabling and university timetabling. In fact, such constraints can force the timetable to be completely filled in, which is a constraint genuinely hard to satisfy.

3.2 Course timetabling

The university course timetabling problem consists in scheduling a set of lectures for each course within a given number of rooms and time periods.

The main difference from the school problem is that university courses can have students in common, whereas school classes are made up of disjoint sets of students (which can be treated as a single entity). If two courses share common students then they conflict, and cannot be scheduled at the same period.

Moreover, school teachers always teach more than one class, whereas in universities, a professor usually teaches only one course.

Finally, in the university problem, availability of rooms (and their size) plays an important role, whereas in the school problem they are often neglected because, in most cases, we can assume that each class has its own room.

3.3 Examination timetabling

The examination timetabling problem requires the scheduling of a given number of exams (one for each course) within a given amount of time in a set of rooms.

The examination timetabling is similar to course timetabling, and it is difficult to make a clear distinction between the two problems. Nevertheless, it is possible to state some broadly-accepted differences between the two problems. Examination timetabling has the following characteristics (differently from the course timetabling problem):

- There is only one exam for each subject.
- The conflict condition is generally strict. In fact, we can accept that a student is forced to skip a lecture due to overlapping, but not that a student skips an exam.
- There are different types of constraints, e.g. at most one exam per day for each student, and not too many consecutive exams for each student.
- The number of periods may vary, in contrast to course timetabling where it is fixed.
- There can be more than one exam per room, but not more than one lecture.

3.4 Discussion

In the previous classification we only sketched the main features of each problem, for the interested reader we refer to [21] for the formal definitions of these problems.

The common ground of these problems is to be all combinatorial problems (search or optimization ones), being NP-hard, and to show similar types of constraints (overlap, availability, capacity, ...). However, most of all share the fact of having been recognized to be genuinely "difficult on the average" in practical real-life cases. For this reason, such problems have not been addressed in a complete and satisfactory way yet, and are still matter of theoretical research and experimental investigation.

Unfortunately, very few benchmark instances have emerged so far from the scientific community. For examination timetabling, a set of instances, the so-called Toronto's benchmarks[3], are available and widely accepted, whereas for the other two no benchmark is available. This situation is probably due to the fact that this is a very practical field, and people working on timetabling are more interested to solve their specific problem variant, rather than comparing results.

4 Local Search for Timetabling Problems

In the following subsections, we present our research on the application of the local search paradigm in the solution of the educational timetabling problem. We briefly give an informal definition of the representation of each problem and we describe the local search setting we have employed.

4.1 Local Search for High-school Timetabling

The high school timetabling problem has been considered in [20]. We briefly report the results of that paper.

Assuming that there are m classes $c_1, \ldots, c_m$, n teachers $t_1, \ldots, t_n$, and p periods $1, \ldots, p$, we represent (as proposed by Colorni *et al.* [11]) a timetable as an integer-valued matrix $M_{n \times p}$

[3]Available from `ftp://ftp.mie.utoronto.ca/pub/carter/testprob`

such that each row j of M represents the weekly assignment for teacher t_j. In particular, each entry m_{jk} contains the index of the class that teacher t_j is meeting at period k. The value $m_{jk} = 0$ represents the fact that t_j is not teaching at period k, whereas values larger than the number of classes m represent special activities, such as being available for temporary teaching posts, teacher-parents meetings, and assisting assignments. The search space is composed by the family of matrices M, including the infeasible ones.

We consider two different neighborhoods. The first is the one that naturally fits in our representation. It is obtained by simply swapping two distinct values in a given row. That is, the lectures of a teacher t in two different periods p_1 and p_2 are exchanged between them, or, in case that one value is 0, one lecture is moved to a different period. We call such a move an *atomic move*, and we identify it by the triple $\langle t, p_1, p_2 \rangle$.

We also consider a more complex move type called *double move*. A double move is made up by a pair of atomic moves, so that the second one "repairs" the infeasibility (or one of the infeasibilities) created by the first one. This is obtained by exchanging the lecture that conflicts with the just inserted one (obviously new infeasibilities can be created).

Other authors have made different choices. For instance, Costa [12] defines as move the reassignment of a single lecture to a different period. However, in his representation, a single teacher can teach more than one lecture at the same time, and therefore a swap of assignments for a single teacher can be done in two consecutive (feasible) moves, with both assignments in the same period at the intermediate step.

Moving to the search strategy, we use a tandem search based on HC and TS, in which the HC uses the double moves and the TS uses the atomic move. In detail, the initial solution is obtained by scheduling the lectures for each teacher randomly, respecting the requirement matrix (or it can be obtained from previous runs, in case of interactive timetabling). Then, the HC starts to work on the random timetable until it cannot find any improvement for a given number of iterations. At this point, the TS starts and goes on until it makes a given number of iterations without improving. The TS makes also use of the shifting penalty mechanism, explained in Section 2.2.

The HC phase is used for two different purposes: First, it generates the initial solution for the TS. In fact, the use of the TS starting from the random solution is too time consuming, and HC instead represents a fast method to generate a reasonably good initial solution (which generally still contains a few infeasibilities). Second, after the TS has given no improvements for a given number of iterations, it is useful to run the HC on the best solution found. The reason for it is twofold: On the one hand, the HC, using double moves, might find improvements that the TS was not able to find at that stage. On the other hand, the HC with double moves, even if it does not improve the solution (which is often the case), makes substantial sideways modifications. Therefore, it "shakes up" the solution before the TS starts again to try to improve it. The idea underlying such a procedure is that after the TS has worked unsuccessfully for a given number of iterations, it is useful to diversify so that the TS can start in a different direction.

4.2 Local Search for Course Timetabling

A solver for Course Timetabling is described in [15] and it is actually in use to generate the real timetable of the Faculty of Engineering of the University of Udine.

There are q courses $C_1, \ldots, C_q$, p periods $1, \ldots, p$, and r rooms $R_1, \ldots, R_r$. Each course C_i consists of l_i lectures to be scheduled in distinct time periods, and it is taken by s_i students.

Each room R_j has a capacity c_j, in terms of number of seats.

The search space of our algorithm is the family of integer-valued $q \times p$ matrices T, such that $T_{ik} = j$ (with $1 \leq j \leq r$) means that course C_i has a lecture in room R_j at period k, and $T_{ik} = 0$ means that course C_i has no class in period k.

The first neighborhood relation we consider is defined by the reschedule of a lecture of the course C_i from the period k_1 to a different period k_2 (leaving the room unchanged). This kind of move, called *ChangeTime*, tries to fix the infeasibilities due to conflicting courses, and unavailabilities of teachers.

The other move type taken into account is called *ChangeRoom* and its effect is to replace the room of a lecture of a given course, without changing the period.. Specifically, the move replaces the room R_j with the room R'_j for the lecture of the course C_i scheduled in the period k. This move type fixes infeasibilities related to room capacity, and room clashes (multiple lectures in one room).

As search strategy, We use a token ring composed of four runners, namely two HC and two TS using the two neighborhood relations (ChangeTime and ChangeRoom). In details, the algorithm start with a random assignment of rooms and periods to the lectures, and performs the runners starting form the HC ones.

As for school timetabling, the HC runners are used mostly for fast diversification, whereas the TS (which spend 90% of the total time) try to improve the solution.

Both the TS runners resulted more effective if they use the shifting penalty mechanism explained in Section 2.2.

4.3 Local Search for Examination Timetabling

In [14] we propose a family of tabu search algorithms for the examination timetabling problem. We consider several formulations of the problem presented in the literature, and we compare our algorithms with previous works on the basis of the Toronto's benchmarks. Specifically, we compare with a set of constructive heuristics developed by Carter *et al* [10], and with two memetic algorithms proposed by Burke and co-workers [8, 7].

The basic formulation of the examination timetabling problem (without side constraints) underlies a graph coloring structure: Each node of the graph represents an examination and there is an edge between two nodes u and v if a student is enrolled in both examination u and v. Thus, associating a color to each legal period, the problem reduces to the well known k-GRAPH COLORING problem: assigning colors to the node of the graph in such a way that no pair of adjacent nodes are assigned the same color. Actually, a more realistic model include an edge-weight function that accounts for the number of students involved in two conflicting examinations and a node-weight function that indicates the number of students enrolled to each examination. This representation can fruitfully be exploited for dealing with higher-order conflicts and room capacity constraints.

Regarding the local search settings, we consider a search space which is composed by all the complete colorings of the graph, including the infeasible ones. Furthermore, in our definition, two states are *neighbors* if they differ for the coloring of a single node. Therefore, a move corresponds to changing the color of one node, and it is identified by a triple $(\langle node \rangle, \langle old\_color \rangle, \langle new\_color \rangle)$.

In order to identify the most promising moves at each iteration, we maintain the so-called *violations list* VL, which contains the examinations that are involved in at least one violation (either hard or soft). A second (shorter) list HVL contains only the nodes that are involved in violations of hard constraints. In different stages of the search , nodes are selected either from VL or from HVL. Nodes not in the lists are never analyzed.

The best algorithm we have implemented is a tandem one that alternates two TS runners. Both runners use the shifting penalty mechanism and they make an exhaustive exploration of the neighborhood. The first runner selects the nodes from the violations list VL, whereas the second one uses an adaptive combination of VL and HVL. Intuitively, the first runner searches for any kind of improvement, whereas the second one focuses of hard constraints. The latter, however, once it has found a feasible solution, automatically expands the neighborhood to include also moves that deal with soft constraints, trying to optimize it. Both algorithms make use also of the shifting penalty algorithm.

The outcome of this algorithm is comparable with the results presented in the literature, both in terms of solution quality and with respect to the computational time required, and it outperforms them in a number of instances.

5 A Local Search Software Tool: EASYLOCAL++

It is worth noticing that writing the local search applications just described for the educational timetabling problem requires a significant programming effort. This is particularly true in a research environment, where there is the need to try out different techniques and, therefore, to develop not only a single algorithm, but rather a family of algorithms among which to choose. On the other hand, also the development process consists of a set of repetitive tasks: for example, the control logic of a tabu search algorithm is always the same, regardless of the underlying optimization problem at hand, and it must be explicitly encoded in any tabu search application with few possibilities of reuse.

Considering these issues from a software engineering point of view, and looking for a support for our research on local search, we designed and implemented EASYLOCAL++: an object-oriented framework that can be used as a general tool for the development and the analysis of local search algorithms in C++.

A framework is a special kind of software library, which consists of a hierarchy of abstract classes. The user defines suitable derived classes, which implement the virtual functions of the framework ones. Frameworks are characterized by the *inverse control* mechanism for the communication with the user code: The functions of the framework call the user-defined ones, and not the other way round. Therefore, a framework provides the full control structure of the invariant part of the algorithms, and the user has only to supply the problem-specific details.

The basic idea behind EASYLOCAL++ is to capture the essential features of most local search techniques and their possible compositions. The core of the framework is composed by a set of cooperating classes that take care of different aspects of local search. The user's application is obtained by writing derived classes for a selected subset of the framework ones; such user-defined classes contain only the specific problem description, but no control information for the algorithm. In fact, the relationships between classes, and their interactions by mutual method invocation, are completely dealt with by the framework.

One of the main characteristics of EASYLOCAL++ is its full reusability: Once the basic data structures and operations are defined and "plugged-in", the system provides for free a straight implementation of all standard local search techniques and a large variety of their combinations. In addition, the system allows the user to generate and to experiment new combinations of features (e.g., neighborhood structures, initial state strategies, and prohibition mechanisms) with a conceptually clear environment and a fast prototyping capability.

EASYLOCAL++ is described in [13] and it is free for download[4]. Its application to course timetabling is provided in [15].

6 Discussion and Conclusions

According to our experience, local search has worked well for all educational timetabling problems. In addition, the search space and the neighborhood relations used are quite intuitive and relatively easy to identify and implement.

Local search techniques, giving the possibility to start the search from any timetable, easily allow for interactive construction and maintenance of timetables. In fact, once a timetable has been generated, it can be used as the starting point for a new search after some constraints have been manually modified. The ability to work interactively is widely recognized as crucial for timetabling systems in the research community. To this respect, local search has a great advantage over other methods, such as constructive ones and genetic algorithms.

Unfortunately, for two of the three problems, the absence of both a common definition and widely-accepted benchmarks prevents us from comparing with other algorithms appearing in the literature. For the only problem for which we could make a limited comparison with previous research, namely examination timetabling, our solvers found better solutions than previous solvers for a number of instances.

References

[1] E. Aarts and J. K. Lenstra. *Local Search in Combinatorial Optimization*. John Wiley & Son, Chichester, 1997.

[2] D. Abramson. Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, 37(1):98–113, 1991.

[3] R. Alvarez-Valdes, G. Martin, and J. Tamarit. Constructing good solutions for the Spanish school timetabling problem. *Journal of the Operational Research Society*, 47, 1996.

[4] R. Battiti. Reactive search: Toward self-tuning heuristics. In V. J. Rayward-Smith, editor, *Modern Heuristic Search Methods*, pages 61–83. John Wiley and Sons Ltd, 1996.

[5] E. Burke and M. Carter, editors. *Proc. of the 2nd Int. Conf. on the Practice and Theory of Automated Timetabling*, number 1408 in Lecture Notes in Computer Science. Springer-Verlag, 1997.

[6] E. Burke and W. Erber, editors. *Proc. of the 3rd Int. Conf. on the Practice and Theory of Automated Timetabling*, number 2079 in Lecture Notes in Computer Science. Springer-Verlag, 2000.

[7] E. Burke and J. Newall. A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

[8] E. Burke, J. Newall, and R. Weare. A memetic algorithm for university exam timetabling. In *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling*, pages 241–250, 1995.

[4]From the web page `http://www.diegm.uniud.it/schaerf/projects/local++`

[9] E. Burke and P. Ross, editors. *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling*, number 1153 in Lecture Notes in Computer Science. Springer-Verlag, 1995.

[10] M. W. Carter, G. Laporte, and S. Y. Lee. Examination timetabling: Algorithmic strategies and applications. *Journal of the Operational Research Society*, 74:373–383, 1996.

[11] A. Colorni, M. Dorigo, and V. Maniezzo. Metaheuristics for high school timetabling. *Computational Optimization and Applications*, 9(3):275–298, 1998.

[12] D. Costa. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76:98–110, 1994.

[13] L. Di Gaspero and A. Schaerf. EASYLOCAL++: An object-oriented framework for flexible design of local search algorithms. Technical Report UDMI/13/2000/RR, Dipartimento di Matematica e Informatica, Università di Udine, 2000. Available at `http://www.diegm.uniud.it/schaerf/projects/local++`.

[14] L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In E. Burke and W. Erben, editors, *Proc. of the 3rd Int. Conf. on the Practice and Theory of Automated Timetabling*, number 2079 in Lecture Notes in Computer Science, pages 104–117. Springer-Verlag, Berlin-Heidelberg, 2001.

[15] L. Di Gaspero and A. Schaerf. Writing local search algorithms using EASYLOCAL++. In S. Voß and D. L. Woodruff, editors, *Optimization Software Class Libraries*, OR/CS. Kluwer Academic Publishers, Boston, 2001. To appear.

[16] M. Gendreau, A. Hertz, and G. Laporte. A tabu search heuristic for the vehicle routing problem. *Management Science*, 40(10):1276–1290, 1994.

[17] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[18] C. C. Gotlieb. The construction of class-teacher timetables. In C. M. Popplewell, editor, *IFIP congress 62*, pages 73–77. North-Holland, 1963.

[19] A. Hertz. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54:39–47, 1991.

[20] A. Schaerf. Local search techniques for large high-school timetabling problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 29(4):368–377, 1999.

[21] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

[22] T. Stützle. Iterated local search for the quadratic assignment problem. Technical Report AIDA-99-03, FG Intellektik, TU Darmstadt, 1998.

[23] J. Thomson and K. Dowsland. General cooling schedules for simulated annealing-based timetabling system. In *Proc. of the 1st Int. Conf. on the Practice and Theory of Automated Timetabling*, pages 345–363, 1995.

[24] S. Voß, S. Martello, I. Osman, and C. Roucairol, editors. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers, 1999.